Amendments to the Specification:

Please amend the abstract and specification as follows:

Please amend the Abstract as follows:

A generic mobile synchronization framework facilitates synchronization of data objects between platforms by comparing these data objects from one platform with a replica of data objects on the other platform. Generic messages identifying the differences in the data objects are converted into an adaptive message suitable for use by the underlying synchronization hardware and sent to the ~~eh~~ platform needing synchronization. That platform converts the adapted messages ~~them~~ to the original generic message, and executes them, synchronizing the data objects in that platform with the originating platform.

Please replace paragraph [004] with the following amended paragraph:

[004] A user typically has both a principal computing platform that he uses as his main computing device to add, change, or delete data and one or more auxiliary computing platforms that he may also use to add, change, or delete data. The principal or auxiliary platforms may run Linux®, Windows®, MacOS®, Symbian®, Windows Mobile PocketPC®, or Palm® operating systems.

Please replace paragraph [005] with the following amended paragraph::

[005] Currently, synchronization requires underlying synchronization software that depends on the specific computing platform needing synchronization. For example, Microsoft

provides Activesync® synchronization software to synchronize data between applications such as Outlook® on principal platforms and auxiliary computing platforms running the Microsoft Windows Mobile PocketPC® operating system.

Please replace paragraph [021] with the following amended paragraph:

[021] FIG. 9 is a flowchart of a PCP-PCP-ACP process ~~process~~ consistent with the present invention.

Please replace paragraph [032] with the following amended paragraph:

[032] ACP 120 may be any type of computing platform running applications with data that may be synchronized with another platform, such as PCP 1~~1~~0. Examples of ACP 120 include a desktop computer, a laptop computer, a notebook computer, a PDA, a handheld computer, or a pocket computer. Although this example involves synchronizing data between a principal and an auxiliary computing platform, the invention is not so limited and can operate in many other environments, such as between two principal or two auxiliary platforms.

Please replace paragraph [034] with the following amended paragraph:

[034] Systems consistent with the principles of the present invention may comprise a generic mobile synchronization framework ("GMSF") that facilitates communication of data between applications and underlying synchronization architecture. Instances of the GMSF are located in both the PCP 110 and the ACP 120. The GMSF provides an application interface module ~~to~~ that interfaces the GMSF to the applications.

Please replace paragraph [045] with the following amended paragraph:

[045] Similar to PCP 110, ACP 120 may be implemented by a general-purpose computer running the appropriate computer programs stored in the computer, or a specially constructed computing platform. ACP 120 may also be implemented with a wide variety of components including, central processing unit 255, I/O interface 260, network interface 265, and display 2050.

Please replace paragraph [049] with the following amended paragraph:

[049] FIG. 3 is a functional block diagram of operations performed by PCP 110 and ACP 120. PCP application 235 in PCP 110 has one or more data objects to be synchronized. Each data object may be assigned a unique data object identifier. For example, a universally unique identifier ("UUID") may be assigned to each data object. A UUID is an identifier that can be used across all computers and networks wherever a unique identifier is required. Such an identifier has a very low probability of being duplicated. For example, in the Windows.NET® framework a UUID may be implemented through a globally unique identifier ("GUID"), which is a 128-bit integer (16 bytes) identifier that serves as a UUID. Because the creation of UUID's in some frameworks, such as the Windows.NET® framework, requires the presence of a network adapter and some ACPs 120 may not feature a network adapter, for example in a Windows Mobile® device, the PCP 110 may generate one or more unassigned UUIDs for transfer to the ACP 120. The ACP 120 may use these received unassigned UUIDs when new data objects are generated in ACP 120.

Please replace paragraph [054] with the following amended paragraph:

[054] FIG. 4 is a diagram of SyncDB 140 and its associated tables consistent with the present invention. The SyncDB can be coupled to one or more SyncStore databases 150a. SyncDB 140 may contain one or more tables used in the synchronizing process. SiteInfo table 410 may contain entries linking User Identifiers 415 to respective Device Identifiers 420. User Identifiers 415 may be unique characters to identify an individual. Device Identifiers 420 may be unique characters to identify an ACP. Each User Identifier 415 may link to one or more Device Identifiers 420. Thus, each user may synchronize multiple devices to a PCP application 235. Settings 422, to be discussed later, may be stored in association with the User Identifiers 415.

Please replace paragraph [056] with the following amended paragraph:

[056] FIG. 5 is a diagram of SyncStore database 150 and its stored data objects consistent with the present invention. During synchronization, data objects from PCP application 235 may be replicated, stored, and grouped together in SyncStore database 150. For example, all data objects 515, 520 supporting Transaction A 510 may be grouped. Similarly, all data objects 530, 535 supporting Transaction B 525 may be grouped. Thus, for example, if an error is generated with respect to any data object association with a transaction, the grouping allows rollback or cancellation of all data object synchronizations related to that transaction.

Please replace paragraph [058] with the following amended paragraph:

[058] PCP application interface module 610 provides the flexibility to interconnect different PCP applications 235 to PCP synchronization framework 240 without requiring an entirely new set of code each time. Instead, only PCP application interface module 610 need be

changed to interface with different applications. For example, a programmer could create one PCP application interface module 610a to connect to one business application and another PCP application interface module 610~~b~~ to connect to another business application, without having to create an entirely new PCP synchronization framework 240. The programmer would merely place the proper objects and methods in place to interface to the appropriate application data objects. This promotes the reuse of code to reduce programming expenses and time as compared to an entire rewrite of most of the framework code.

Please replace paragraph [059] with the following amended paragraph:

[059] Settings module 630 may permit the user, via PCP application interface module 61~~3~~0, to select various settings and parameters for synchronization. These settings may then be stored in a database, such as SyncDB, and associated with a User Identification 415 for the user. Settings may include ACPs associated with the user of the application or PCP, specialized delta calculators (explained above), categorization of data objects, and synchronization settings for categories.

Please replace paragraph [063] with the following amended paragraph:

[063] Methods and objects in SyncStore database 150 may include: StoreUser (the User Identifier associated with the object); TargetSiteID (the Device Identifier for the ACP intended to be synchronized); synchronization type (Normal Synchronization, Data Recovery [~~(~~skipping the delta process and pushing data to the ACP from the previous replica regardless of any data changes in the PCP]~~)~~, or Forced Synchronization [~~(~~skipping the delta process and pushing data to the ACP from the PCP regardless of the content of the replica]~~)~~); Begin Transaction (noting the

beginning of a series of data objects for a transaction); End Transaction (noting the end of a series of data objects for a transaction; one or more data objects); SyncNow (to initiate synchronization); and Status.

Please replace paragraph [066] with the following amended paragraph:

[066] Once delta services module 652 computes the differences in the data objects, message builder module 654 may generate generic messages, in conjunction with query builder module 6~~8~~80, to implement the differences on data objects in ACP 120. Messages may be built in a generic fashion, such as an SQL statement, or may be embedded within XML (extensible Markup Language) data. By using XML, metadata may be included in the messages to inform the recipient of the messages about not only the data itself, but also the data structure, i.e., fields, types, target fields, etc.

Please replace paragraph [070] with the following amended paragraph:

[070] FIG. 7 is a functional block diagram illustrating an ~~in~~ ACP synchronization framework 285 consistent with the present invention. Because, in this embodiment, most of the work of the synchronization takes place on PCP 110, which is generally where the heaviest computing power is located, ACP synchronization framework 285 is much simpler. ACP synchronization framework 285 receives inbound messages from PCP 110 in the inbound queue and syncserver ("IQS") module 720. IQS module 720 executes the inbound messages on the data objects of the associated ACP application 280. Any errors or generated confirmations enter outbound queue module 710. Outbound queue module 710 also contains any data objects that

may have been added or changed since the last synchronization, and may pass generic messages relaying these data objects to ACP synchronization adapter 290 for transfer to PCP 110.

Please replace paragraph [072] with the following amended paragraph:

[072] FIG. 8 is a flowchart of a GMSF~~P~~ consistent with present invention. The GMSP may include PCP GMSP 805 and ACP GMSF~~P~~ 825. PCP GMSP 805 may comprise a PCP-PCP-ACP process ("PPA") 810 and a PCP-ACP-PCP ("PAP") process 820. ACP GMSF~~P~~ 825 may comprise an ACP-PCP-ACP process ("APA") 830 and an ACP-ACP-PCP ("AAP") process 840. PPA process 810 sends synchronization messages to APA process 830 for execution. AAP process 840 sends synchronization messages to AAP process 820 for execution.

Please replace paragraph [074] with the following amended paragraph:

[074] Syncstore module 640 stores the fetched data objects ~~by~~ (stage 910) and delta service module 652 compares them to a replica data set reflecting the last synchronization state of ACP 120 (stage 915). These actions generate a delta set of changes.

Please replace paragraph [076] with the following amended paragraph:

[076] FIG. 10 is a flowchart of the "fetch data process" of stage 905 in FIG. 9 consistent with the present invention. Using selection module 620, the user may select data objects from PCP application 235 for synchronization ~~by~~ and one or more ACPs (stage 1005) (via their respective IDs). Also, synchronization settings may be set using settings module 630, as previously explained (stage 1010).

Please replace paragraph [081] with the following amended paragraph:

[081] "Data recovery synchronization" may be used if the ACP data objects are all lost in order to recover the ACP data objects to their last known state if ~~is~~ desired. Similar to "forced synchronization," "data recovery synchronization," may bypass the delta processor and force a copy of the replica database to the ACP. The user may select the type of synchronization by the settings, as previously described. The data objects in the SyncStore are then set to "created" by syncstore module 640 (stage 1120).

Please replace paragraph [082] with the following amended paragraph:

[082] FIG. 12 is a flowchart of the "compare process" of stage 915 in FIG. 9 consistent with the present invention. Syncengine module 650 may perform this process. Module 650 creates a replica of data objects in SyncStore 150 (stage 1210). This replica becomes flagged as the new replica of the data objects in ACP 120 because the new replica should be the same as the SyncStore data objects if~~, assuming~~ the synchronization is processed without error~~,~~ (stage 1220). The previous new replica is subsequently reflagged as the old replica (stage 1230). The old replica and the SyncStore data objects pass to the delta services module 652 for comparison (step 1240).

Please replace paragraph [083] with the following amended paragraph:

[083] FIG. 13 is a flowchart of a standard delta generation process of ~~at~~ stage 915 in FIG. 9 consistent with the present invention. The illustrated delta generation process may be for the default delta generation process in delta services module 652. Upon entry of the delta generation process, an initial check is made to see if a plug-in or substitute delta generation

process has been selected for these data objects (stage 1302). If so, the plug-in delta generation process is executed in place of the standard one.

Please replace paragraph [084] with the following amended paragraph:

[084] Assuming the standard delta generation process 1250 ~~915~~ is to be used, Delta Add Objects are created for each data object in the SyncStore (stage 1305). Initially the delta generation process begins by assuming that all data objects in the SyncStore must be added to the ACP application 280's data objects. Process 1250 ~~915~~ then begins to check each data object in the old replica against each data object in SyncStore 150 (stage 1310). If process 1250 ~~915~~ does not find the old replica data object in a Delta Add Object (stage 1315), PCP application 235 has likely deleted the old replica data object (stage 1320). If so, the Delta Add Object changes to a Delta Delete Object to indicate that this data object needs to be deleted from ACP application 280's data objects (stage 1325).

Please replace paragraph [085] with the following amended paragraph:

[085] If process 1250 ~~915~~ finds the data object from the old replica in the Delta Add Objects, it compares the data object in the matching Delta Add Object and the old replica data object (stage 1330). If the data objects are the same (stage 1335), the Delta Add Object is removed (stage 1340). Otherwise, the Delta Add Object is changed to a Delta Change Object with updated data object values (stage 1345). Thus, the results of delta generation process 1250 ~~915~~ may be Delta Add Objects, Delta Delete Objects, or Delta Change Objects.

Please replace paragraph [089] with the following amended paragraph:

[089] The remaining processes are described more briefly because the details of all are similar to the PPA process. FIG. 15 is a flowchart of APA process 830 consistent with the present invention. Using ACP synchronization adapter 290, the process receives adapted messages from ACP 120 via the underlying synchronization software (stage 1510). The ACP synchronization adapter 290 converts adapted messages into generic messages (stage 1520). IQS module 720 may then execute the generic messages on the data store of ACP application 280 (stage 1530). If any errors occur upon such execution, IQS module 720 places the errors entering an ACP outbound queue for relaying back to PCP application 235 (stage 1540).